ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

(Laboratorio di)
Amministrazione di sistemi

Utenti e file

Marco Prandini

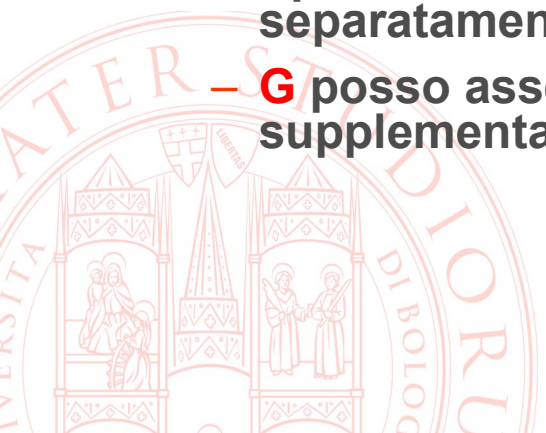
Dipartimento di Informatica – Scienza e Ingegneria

Utenti

- Gli utenti sono i *soggetti* di tutte le operazioni svolte sul sistema, utilizzati per determinare i permessi di accesso a qualsiasi risorsa (*oggetto*)
- Ogni utente **deve** appartenere a un gruppo
  - al login l'utente si trova a operare come membro di tale gruppo
- Ogni utente **può** appartenere a un numero arbitrario di gruppi supplementari
  - durante una sessione di lavoro, l'utente può liberamente assumere l'identità di qualsiasi gruppo del quale sia membro

useradd

- useradd è lo standard per creare nuovi utenti, ha una granularità molto fine ed è molto utile per automatizzare tale processo
- I valori predefiniti per le caratteristiche dell'utente creato sono impostati nel file /etc/login.defs
- Parametri principali
  - m crea la home del utente, usa come template i files dentro /etc/skel
  - s assegna la shell all'utente, le possibili shell sono indicate dentro il file /etc/shells, altrimenti prende il default
  - U crea un gruppo con lo stesso nome dell'utente
  - K con questo parametro è possibile specificare la UMASK=0077
  - p dopo questo parametro è possibile inserire la password utente MA come riportato nella man page è sconsigliato, molto meglio usare passwd separatamente
  - G posso assegnare l'utente all'atto della creazione ad un gruppo supplementare esempio sudo



Il db degli utenti

- Le credenziali locali sono in
  - `/etc/passwd`, world-readable, una riga per utente:
    `prandini:x:500:500:Marco Prandini:/fat/home:/bin/bash`
  - `/etc/shadow`, accessibile solo a root, linee corrispondenti a **passwd**
    `prandini:$1$/PBy29Md$kjC1F8dvHxKhnvMTWelnX/:12156:0:99999:7:::`
  - Nota: non rimuovere il segnaposto 'x' nel secondo campo di passwd, o il sistema non guarderà il file shadow e non riconoscerà la password
- L'appartenenza ai gruppi è l'unione dell'informazione presente in **/etc/passwd** riguardante il gruppo principale di ogni utente e del contenuto dei file
  - `/etc/group`, world-readable, una riga per gruppo:
    `sudo:x:27:prandini`
  - `/etc/gshadow`, accessibile solo a root, linee corrispondenti a **group**
    `sudo:*::prandini`
- Il comando **id <USER>** riporta tutte le informazioni di identità

Gestione di utenti esistenti

- Il comando **usermod** permette di modificare, coi suoi diversi parametri, tutte le caratteristiche dell'utente
  - come useradd, può essere usato solo da root
- Esiste anche una serie di comandi specifici per cambiare singole proprietà
  - possono essere invocati da root per gestire qualsiasi utente
  - possono essere invocati anche da utenti standard per agire ovviamente solo sul proprio account

chsh modifica della shell di login

chfn modifica del nome reale

passwd modifica della password

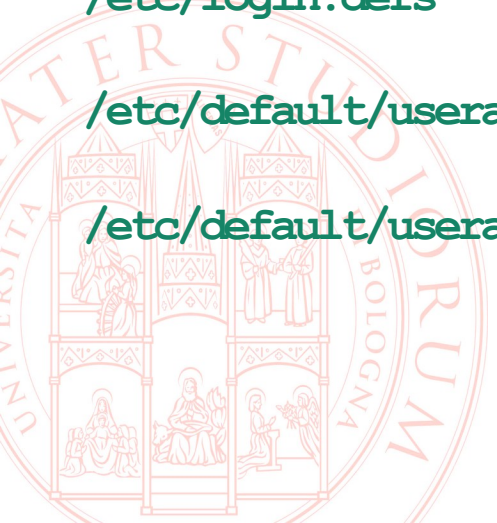
Età delle password

- Il file shadow contiene dati sulla validità temporale della password, esaminabili e modificabili con **chage**:

```
<name>:<pw>:<date>:PASS_MIN_DAYS:PASS_MAX_DAYS:PASS_WARN_AGE:INACTIVE:EXPIRE:
```

- Significato e nome del file da cui viene preso il valore di default::

/etc/login.defs	PASS_MAX_DAYS	Maximum number of days a password is valid.
/etc/login.defs	PASS_MIN_DAYS	Minimum number of days before a user can change the password since the last change.
/etc/login.defs	PASS_WARN_AGE	Number of days when the password change reminder starts.
/etc/default/useradd	INACTIVE	Number of days after password expiration that account is disabled.
/etc/default/useradd	EXPIRE	Account expiration date in the format YYYY-MM-DD.

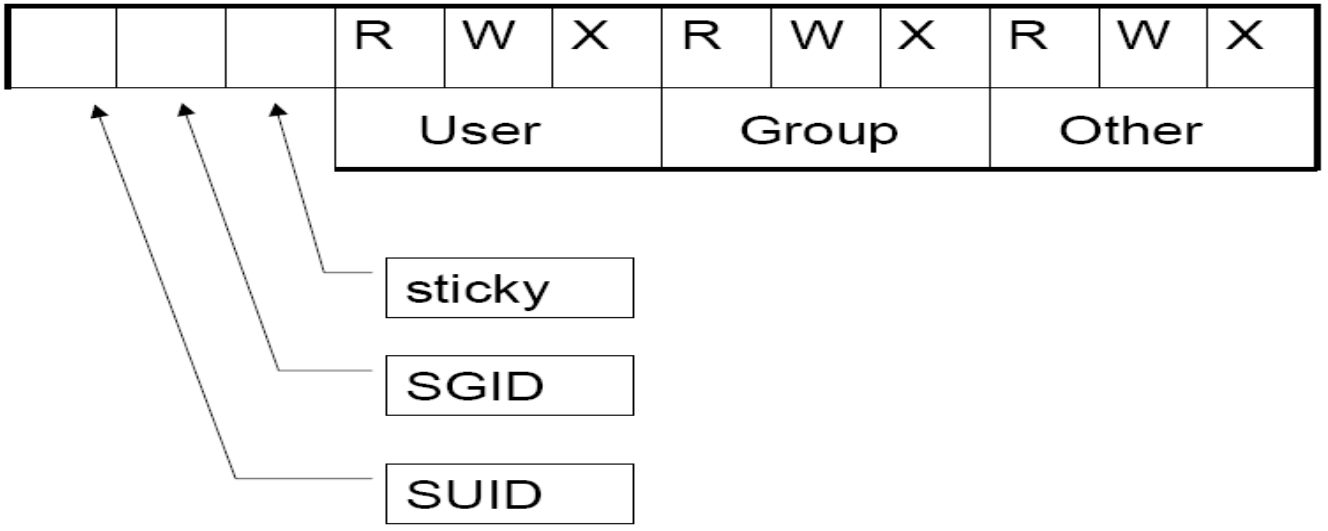


Altri comandi di creazione e gestione

- **adduser** è uno script in Perl per creare un nuovo utente non è lo standard in tutte le distribuzioni, è presente in Debian, Ubuntu
  - il programma chiede i dettagli interattivamente
  - utile quindi se usato in maniera estemporanea, molto poco invece se abbiamo bisogno di scriptare il processo di creazione utenti
- Per la creazione di gruppi, esistono analogamente **groupadd** e **addgroup**
- Altri comandi utili:
  - **gpasswd** modifica password e lista utenti di un gruppo
  - **getent** interroga il db utenti o gruppi
  - **last** elenca i login effettuati sul sistema
  - **lastlog** mostra la data di ultimo login di ogni utente
  - **faillog** mostra i login falliti sul sistema

Autorizzazioni su Unix Filesystem

- Ogni file (regolare, directory, link, socket, block/char special) è descritto da un i-node
- Un set di informazioni di autorizzazione, tra le altre cose, è memorizzato nell'i-node
  - (esattamente un) utente proprietario del file
  - (esattamente un) gruppo proprietario del file
  - Un set di 12 bit che rappresentano permessi standard e speciali



Significato dei bit di autorizzazione

- Leggermente diverso tra file e directory, ma in gran parte deducibile ricordando che
  - Una directory è semplicemente un file
  - Il contenuto di tale file è un database di coppie (nome, i-node)

R = read (lettura del contenuto)

Lettura di un file

Elenco dei file nella directory

W = write (modifica del contenuto)

Scrittura dentro un file

Aggiunta/cancellazione/rinomina di file in una directory

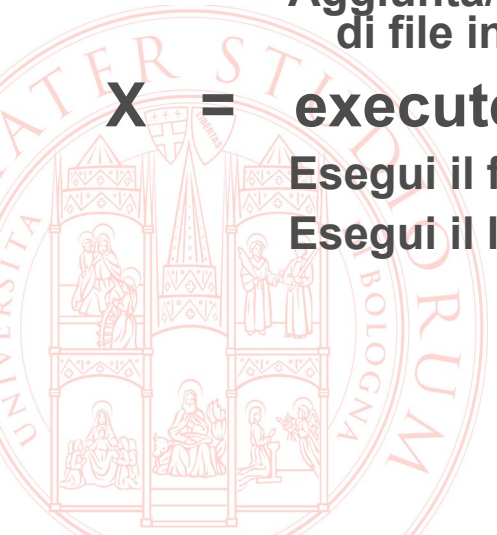
X = execute

Esegui il file come programma

Esegui il lookup dell'i-node nella

NOTA che il permesso 'W' in una directory consente a un utente di cancellare file sul contenuto dei quali non ha alcun diritto

NOTA: l'accesso a un file richiede il lookup di tutti gli i-node corrispondenti ai nomi delle directory nel path → serve il permesso 'X' per ognuna, mentre 'R' non è necessario



Assegnazione dell'ownership

- Alla creazione di un file
  - l'utente creatore è assegnato come proprietario del file
  - Il gruppo attivo dell'utente creatore è assegnato come gruppo proprietario
    - Default = gruppo predefinito, da /etc/passwd
    - L'utente può rendere attivo nella sessione un altro tra i propri gruppi con newgrp
    - Può cambiare automaticamente nelle directoy con SGID settato (vedi seguito)
- Successivamente
  - Comando chown [new_owner]:[new_group] <file>
    modifica owner e/o group owner del file
  - Comando chgrp [new_group] <file>
    modifica group owner del file
    - comunque solo tra quelli di cui l'utente è membro
  - Per entrambi l'opzione -R attiva la ricorsione su cartelle

Assegnazione dei permessi

- Alla creazione: permessi = “tutti quelli sensati” tolta la umask
  - “tutti quelli sensati” significa due cose diverse:
    - `rw-rw-rw-` (666) per i file, l’eseguibilità è un’eccezione
    - `rwxrwxrwx` (777) per le directory, la possibilità di entrarci è la regola
    - la *umask* quindi può essere unica: una maschera che toglie i permessi da non concedere
  - poiché in Linux il gruppo di default group di un utente contiene solo l’utente stesso, una umask sensata è 006 (toglie agli “other” lettura e scrittura)
    - È un settaggio utile per collaborare, crea file manipolabili da tutti i membri del gruppo, a patto che questo sia settato correttamente
  - col comando umask si può interrogare e settare interattivamente nella sessione corrente, per rendere persistente la scelta si usano i file di configurazione della shell

Assegnazione dei permessi

- Successivamente, **chmod** è usato per modificare i permessi
  - Modo numerico (base ottale):

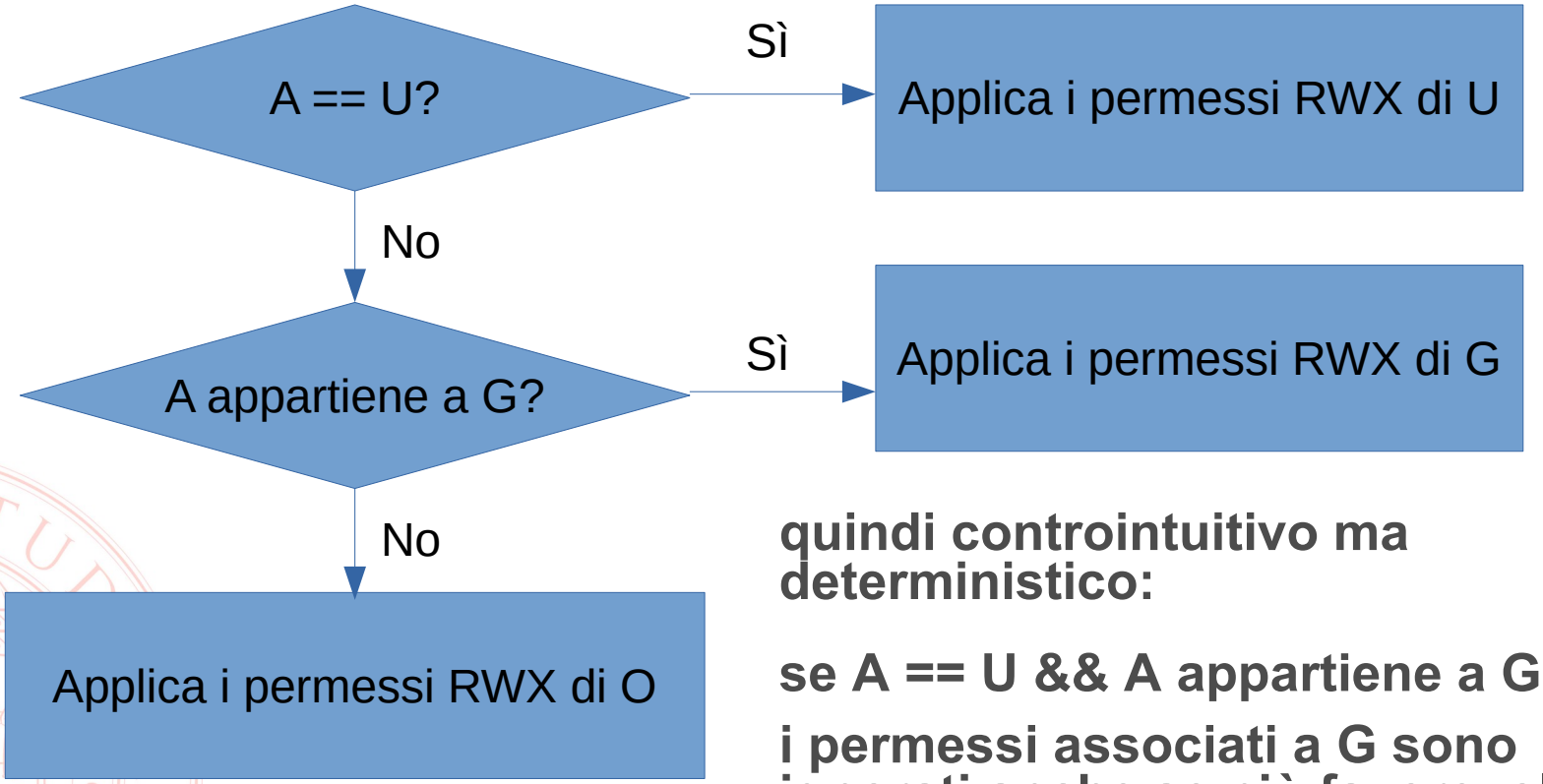
```
chmod 2770 miadirectory
2770 octal = 010 111 111 000 binary = SUID SGID STICKY rwx rwx ---

chmod 4555 miocomando
4555 octal = 100 101 101 101 binary = SUID SGID STICKY r-x r-x r-x
```

  - Modo simbolico:

```
chmod 'a=r,g-rx,u+rw' file
```

Composizione dei permessi

- Quando un utente “A” vuole eseguire un’operazione su di un file, il sistema operativo controlla i permessi secondo questo schema:



quindi controintuitivo ma deterministico:

se A == U && A appartiene a G

i permessi associati a G sono ignorati anche se più favorevoli

SUID e SGID

- Supponiamo che un utente U, che in in dato momento ha come gruppo attivo G, lanci un programma
- Il processo viene avviato con una quadrupla di identità:
  - real user id (ruid) = U
  - real group id (rgid) = G
  - effective user id (euid) identità assunta dal processo per operare come soggetto diverso da U
  - effective group id (egid) identità di gruppo assunta dal processo per operare come soggetto diverso da G
- Normalmente euid=ruid e egid=rgid
- Alcuni permessi speciali attribuiti a file eseguibili possono fare in modo che euid e/o egid siano diversi dai corrispondenti ruid / rgid
  - si definiscono programmi Set-User-ID o Set-Group-ID

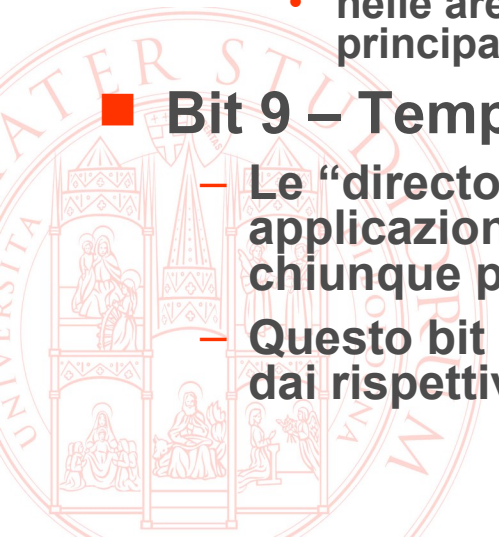
Bit speciali / per i file

I tre bit più significativi della dozzina (11, 10, 9) configurano comportamenti speciali legati all'utente proprietario, al gruppo proprietario, e ad altri rispettivamente

- BIT 11 – SUID (Set User ID)
  - Se settato a 1 su di un programma (file eseguibile) fa sì che al lancio il sistema operativo generi un processo che esegue con l'identità dell'utente proprietario del file, invece che quella dell'utente che lo lancia
- BIT 10 – SGID (Set Group ID)
  - Come SUID, ma agisce sull'identità di gruppo del processo, prendendo quella del gruppo proprietario del file
- BIT 9 – STICKY
  - OBSOLETO, suggerisce al S.O. di tenere in cache una copia del programma

Bit speciali / per le directory

- Bit 11 per le directory non viene usato
- Bit 10 – SGID
  - Precondizioni
    - un utente appartiene (anche) al gruppo proprietario della directory
    - il bit SGID è impostato sulla directory
  - Effetto:
    - l'utente assume come gruppo attivo il gruppo proprietario della directory
    - I file creati nella directory hanno quello come gruppo proprietario
  - Vantaggi (mantenendo umask 0006)
    - nelle aree collaborative il file sono automaticamente resi leggibili e scrivibili da tutti i membri del gruppo
    - nelle aree personali i file sono comunque privati perché proprietà del gruppo principale dell'utente, che contiene solo l'utente medesimo
- Bit 9 – Temp
  - Le "directory temporanee" cioè quelle world-writable predisposte perché le applicazioni dispongano di luoghi noti dove scrivere, hanno un problema: chiunque può cancellare ogni file
  - Questo bit settato a 1 impone che nella directory i file siano cancellabili solo dai rispettivi proprietari



Comandi utili per lavorare coi file

- Elenco e navigazione
- Analisi dei metadati
- Trasferimento dati
- Ricerca nel filesystem
- Archiviazione e compressione

Navigazione

- `pwd` mostra la directory corrente di lavoro
- `cd` permette di spostarsi a un'altra directory
  - esplicitamente nominata, oppure
  - la home dell'utente se invocato senza parametri, oppure
  - la directory in cui ci si trovava prima dell'ultimo cd se invocato con `-`
- ricordiamo che in ogni directory D sono sempre presenti due sottodirectory
  - `.` che coincide con la directory D stessa
  - `..` che coincide con la directory superiore (in cui D è contenuta)

Opzioni principali di ls

- `-l` abbina al nome le informazioni associate al file
- `-a` non nasconde i nomi dei file che iniziano con .
  - per convenzione i file di configurazione iniziano con un punto, non essendo interessanti per l'utente non sono mostrati di default da ls
- `-A` come -a ma esclude i file particolari `.` e `..`
- `-F` pospone il carattere `*` agli eseguibili e `/` ai direttori
- `-d` lista il nome delle directory senza listarne il contenuto
  - il comportamento di default di ls quando riceve come parametro una directory è di elencarne il contenuto, cosa spesso indesiderabile quando nomi di file e directory vengono espansi dalla shell a partire da wildcard
- `-R` percorre ricorsivamente la gerarchia
- `-i` indica gli i-number dei file oltre al loro nome
- `-r` inverte l'ordine dell'elenco
- `-t` lista i file in ordine di data/ora di modifica (dal più recente)

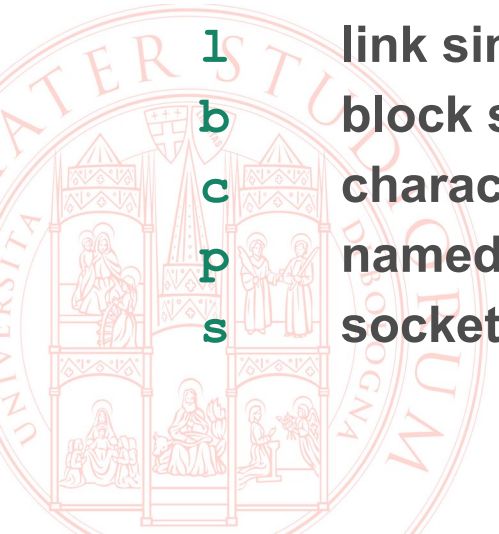
I metadati principali mostrati da ls -l

```
vagrant@bullseye:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1514 Mar 29 11:07 /etc/passwd
```

tipo | **permessi** | **n. link** | **utente e gruppo** | **dim.**

data modifica
si noti che se l'ultima modifica è avvenuta oltre un anno fa, verrà mostrato l'anno invece che l'ora

- **tipi:**
  - `-` file standard
  - `d` directory
  - `l` link simbolico
  - `b` block special (device)
  - `c` character special (device)
  - `p` named pipe (FIFO)
  - `s` socket

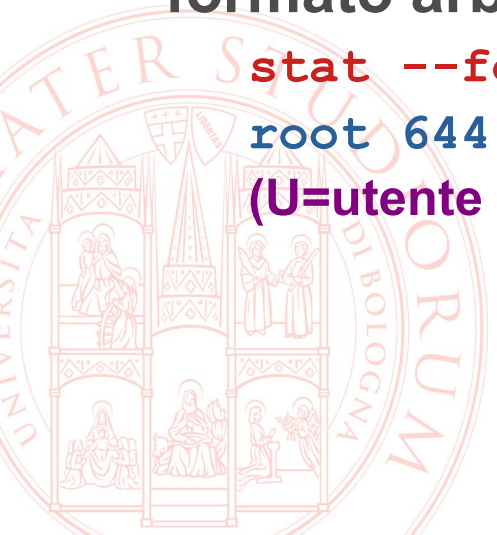


Le marcature temporali (timestamp)

- Ogni file ha tre (o quattro) timestamp distinti
  - mtime modification time istante dell'ultima **modifica del contenuto**
  - atime access time istante dell'ultimo **accesso al contenuto**
  - ctime change time istante dell'ultima **modifica ai metadati**
  - wtime birth time istante della **creazione** del file, **se supportato dal filesystem**
- Queste informazioni vengono gestite automaticamente dal filesystem, ma possono essere cambiate a mano col comando **touch**
- Tutti i metadati possono essere estratti e visualizzati in un formato arbitrario col comando **stat**

```
stat --format='%U %a %z' /etc/passwd
root 644 2021-03-15 08:33:06.381876582 +0100
```

(U=utente proprietario, a=permessi, z=ctime)



Creazione e rimozione di file

- `rm` cancella un file o, meglio, rimuove il link
  - "garbage collection" - il file viene cancellato quando il link count = 0
  - link count = n. link sul filesystem + n. open file descriptors
- `cp` copia un file o più file in una directory
  - attenzione ai file speciali: copio il "concetto" o il contenuto?
- `mv` sposta un file o più file in una directory
- `ln` crea un link ad un file
  - hardlink di default, solo all'interno dello stesso FS e non verso directory
  - symlink con l'opzione `-s`, nessuna limitazione
- `mkdir` crea una directory
- `rmdir` cancella una directory
  - deve essere vuota
  - `rm -r` cancella ricorsivamente

Ricerca nel filesystem con find

- find ricerca in tempo reale
  - quindi esplorando il filesystem → attenzione al carico indotto!

  i file che soddisfano una combinazione di criteri, ad esempio:
  - nome che contenga una espressione data
  - timestamp entro un periodo specificato
  - dimensione compresa tra un minimo e un massimo
  - tipo specifico (file, dir, link simbolici, …)
  - di proprietà di un utente o di un gruppo specificati (o “orfani”)
  - permessi di accesso specificati

  e molti altri
- Esempio:
  - ricercare sotto /usr/src tutti i file che finiscono per .c, hanno dimensione maggiore di 100K, ed elencarli sullo standard output:

```
find /usr/src -name '*.c' -size +100k -print
```

Esecuzione di operazioni sui file trovati

- Una delle opzioni più potenti di find permette, per ciascun oggetto individuato secondo i criteri impostati, di invocare l'esecuzione di un comando:
- Es. mostra il contenuto dei file trovati

```
find /usr/src -name '*.c' -size +100k -exec cat {} \;
```

  - il comando che segue `-exec` viene lanciato per ogni file trovato
  - la sequenza `{}` viene sostituita di volta in volta con il nome del file
  - `\;` è necessario per indicare a find la fine del comando da eseguire

- Es. elenca solo i file regolari "orfani" modificati meno di due giorni (2*24 ore) fa che contengono TXT

```
find / -type f -nouser -mtime -2 -exec grep -l TXT {} \;
```

Ricerca di file con locate

- **locate** effettua la ricerca su di un database indicizzato
  - Il database deve essere aggiornato periodicamente con l'utility updatedb
- Vantaggi su find
  - Carico sul sistema ridotto a una singola esplorazione per ogni periodo, indipendentemente dal numero di query successive
  - Esplorazione pianificabile nei momenti di basso carico
  - Risposta pressoché istantanea
- Svantaggi rispetto a find
  - Unico criterio di ricerca: pattern nel nome
  - Risposte potenzialmente obsolete
    - file creati dopo l'esplorazione non vengono riportati
    - file cancellati dopo l'esplorazione sembrano ancora esistere

Identificazione del contenuto di file

- In Linux, le estensioni dei nomi hanno come unico utilizzo quello di renderli più leggibili all'utente
- Si può ottenere manualmente l'identificazione con **file**
  - test 1: usa stat per capire se il file è vuoto o speciale
  - test 2: usa il database dei **magic number** per identificare il file
  - test 3: usa metodi empirici per capire se è un file di testo, e in tal caso quale sia la lingua naturale o linguaggio di programmazione

I due formati dei file di testo

- nei sistemi UNIX le linee sono terminate da un carattere:
  - line feed o LF o \n o 0x0A
- nei sistemi DOS/Windows le linee sono terminate da due caratteri:
  - carriage return line feed o CRLF o \r\n o 0x0D0A

- senza conversione opportuna
  - file di origine DOS, su sistemi UNIX hanno caratteri extra a fine linea
    - comunemente visualizzati dagli editor come ^M
    - possono causare errori negli script e nei file di configurazione
  - file di origine UNIX, su sistemi DOS confondono le linee

- strumenti Linux
  - alcuni protocolli di rete convertono automaticamente
  - command line Ubuntu: pacchetto tofrodos → comandi `todos` / `fromdos`
  - nomi alternativi su altre distro, es. `unix2dos` / `dos2unix`

Trasferimento di dati da/per device (locali)

- I comandi più ovvi non sono pratici per trasferire dati da/verso file speciali
  - **cat** e ridirezioni sono utilizzabili in modo "tutto o niente"
  - **cp** non è utilizzabile
- **dd** permette di leggere byte da qualsiasi file (`if=<NOME>`) e scrivere su qualsiasi file (`of=<NOME>`)
  - se **NOME** = **-** si intende STDIN (per `if`) o STDOUT (per `of`)

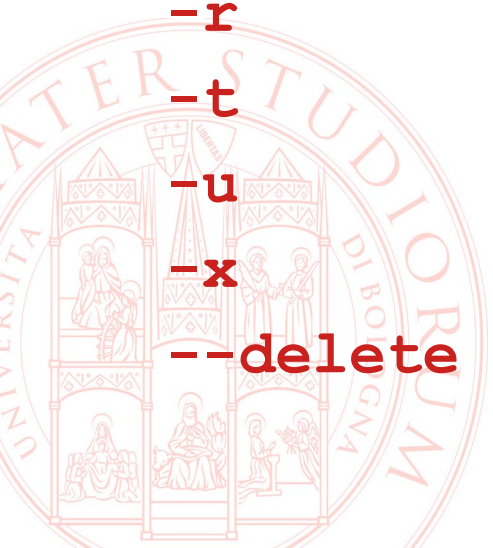
  specificando
  - da che punto iniziare a leggere `skip=<N>`
  - in che punto iniziare a scrivere `seek=<N>`
  - quanti dati trasferire `count=<N>`
  - con che dimensione di blocco operare `bs=<N>`

  inoltre può eseguire trasformazioni di formato e tracciare il progresso del trasferimento

Archiviazione di file

- Per poter agevolmente memorizzare e trasferire una molteplicità di file, eventualmente senza perdere le proprietà associate a ciascuno (ownership, permessi, timestamps…) è comune avvalersi di **tar**. La sintassi prevede che debba essere specificato esattamente uno dei seguenti comandi:

`-A`	concatena più archivi
`-c`	crea un nuovo archivio
`-d`	trova le differenze tra archivio e filesystem
`-r`	aggiunge file ad un archivio
`-t`	elenca il contenuto di un archivio
`-u`	aggiorna file in un archivio
`-x`	estrae file da un archivio
`--delete`	cancella file da un archivio



Archiviazione di file

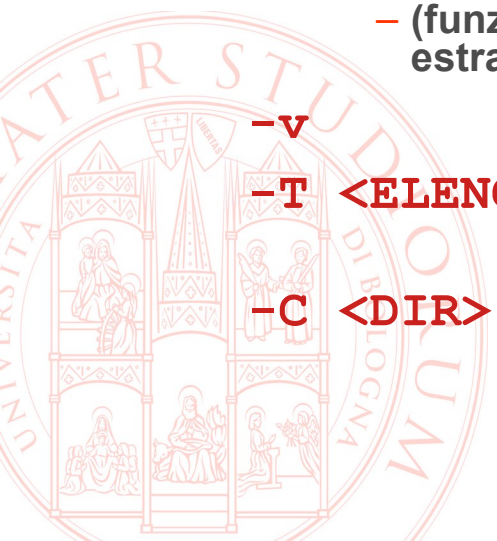
- Le origini di **tar** risalgono ai tempi dei nastri magnetici (il nome è acronimo di Tape ARchiver) quindi di default assume che l'archivio sia su **/dev/tape**.
- L'opzione **`-f <FILENAME>`** viene quindi sempre usata per specificare un file di archiviazione.
  - Dove sensato, **FILENAME** può essere **–** in per indicare
    - lo standard input da cui leggere un archivio con **`d`**, **`t`**, **`x`**
    - lo standard output su cui scrivere l'archivio con **`c`**
- Altre opzioni comunemente usate sono:

`-p` (preserve) conserva tutte le informazioni di protezione
  - (funziona pienamente solo per root, un utente standard quando ricrea i file estraendoli da un archivio è forzato a dargli la sua ownership)

`-v` stampa i dettagli durante l'esecuzione

`-T <ELENCO>` prende i nomi dei file da archiviare da ELENCO invece che come parametri sulla riga di comando

`-C <DIR>` svolge tutte le operazioni come dopo **`cd DIR`**



Archiviazione di file

- Esempi (si noti che il trattino per indicare le opzioni può essere omesso fintanto che non è necessario utilizzare più di un'opzione che richiede parametri)
- creazione

```
tar cvpf users.tar /home/*
```

  - la barra iniziale verrà rimossa in modo da rendere relativi tutti i path
- estrazione

```
tar -C /newdisk -xvpf users.tar
```

  - poiché i path nell'archivio sono relativi, la directory home viene ricreata dentro /newdisk e tutta la gerarchia sottostante viene ricostruita
- pipeline

```
tar cvpf - /home/* | tar -C /newdisk -xvpf -
```

Compressione di file

- tar non comprime
- esistono moltissimi formati di compressione
  - https://linuxhint.com/top_10_file_compression_utilities_on_linux/
  - https://en.wikipedia.org/wiki/List_of_archive_formats
- I più comuni nei sistemi Linux sono
  - estensione `.gz` comando base: `gzip`
  - estensione `.bz2` comando base: `bzip2`
  - estensione `.xz` comando base: `xz`
- Il comando base prende come argomento un file e lo comprime aggiungendo l'estensione
  - con l'opzione `-d` decomprime ricreando il file e rimuovendo l'estensione
  - con l'opzione `-c` riversa il risultato su STDOUT invece che su file
    - filtro!
    - es: `tar cf - * | xz -c > archive.tar.xz`

Compressione di file - scorciatoie

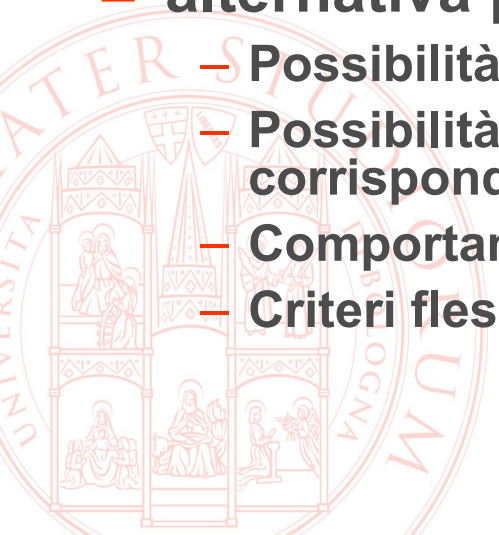
- Esiste tipicamente un comando di decompressione equivalente al comando base invocato con -d
  - es. `gunzip`, `bunzip2`, `unxz`
- Esistono alias per le combinazioni più comuni di filtro di decompressione e comandi di trattamento testo
  - `zcat file.gz` == `gzip -dc file.gz`
  - `zegrep <REGEX> file.gz` == `gzip -dc file.gz | egrep <REGEX>`
  - (idem per i decompressori `bz*`, `xz*`, e per i comandi `*diff`, `*less`, `*cmp`)
- **tar** in particolare supporta opzioni per invocare direttamente la (de)compressione di un archivio

`-z`	usa `gzip`	estensione `.tar.gz` o `.tgz`
`-j`	usa `bzip2`	estensione `.tar.bz2` o `.tbz2`
`-J`	usa `xz`	estensione `.tar.xz` o `.txz`

  - esempio precedente == `tar cJf archive.tar.xz *`

Copia massiva di file (anche remota)

- Il trasferimento di gerarchie di file e cartelle, contenenti file non standard non è gestito correttamente da tutte le versioni di `cp -a` e `scp -R`
- `tar` archivia correttamente tutti i metadati
  - prima possibilità:
    - creare un archivio
    - (eventualmente trasferirlo con scp su un altro host)
    - estrarlo nella cartella di destinazione
- alternativa più evoluta: `rsync`
  - Possibilità di non trasferire file già presenti a destinazione
  - Possibilità di trasferire solo le differenze tra un file sorgente e il corrispondente file a destinazione
  - Comportamento con file speciali configurabile
  - Criteri flessibili di inclusione ed esclusione



Copia massiva di file con rsync

- Sintassi base del comando client

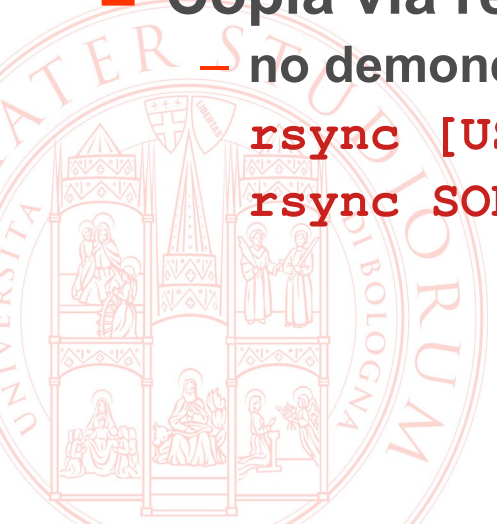
  `rsync [OPZIONI] SORGENTE DESTINAZIONE`
- Copia locale (e remota dove usati negli esempi seguenti)
  - SORGENTE = elenco di file e cartelle
  - DESTINAZIONE = cartella
- Copia via rete con protocollo nativo
  - da / verso host su cui gira il demone rsyncd

    `rsync [USER@]HOST::SRCDIR DESTINAZIONE`

    `rsync SORGENTE [USER@]HOST::DESTDIR`
- Copia via rete via SSH
  - no demone rsyncd richiesto

    `rsync [USER@]HOST:SRCDIR DESTINAZIONE`

    `rsync SORGENTE [USER@]HOST:DESTDIR`



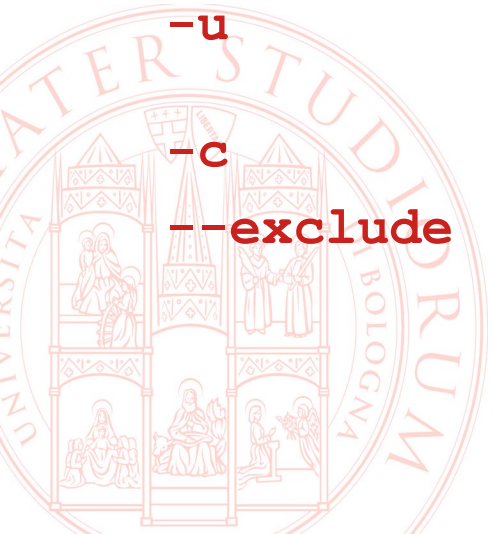
Alcune opzioni di rsync

- Come copiare

`-l` / `-L`	copia i link come link / come file puntato
`-p` / `-o` / `-g`	preserva i permessi, il proprietario, il gruppo
`-t` / `-a` / `-N`	preserva i ts di modifica / accesso / creazione
`-D`	preserva i file speciali
`-a`	= `-rlptgoD`
`-b`	backup (come? `--backup-dir` / `--suffix`)

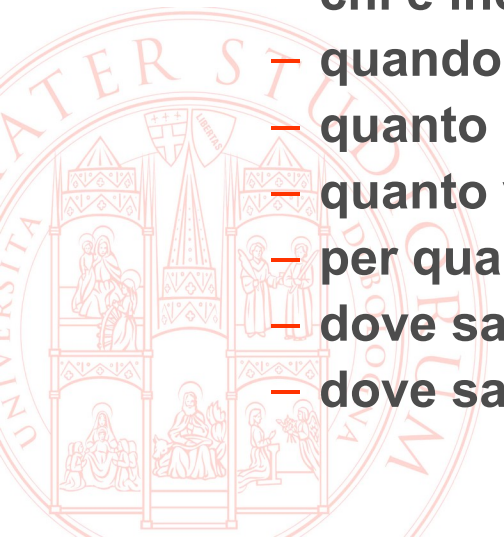
- Cosa copiare

`-r`	ricorsivo
`-u`	salta i file che sono più nuovi a destinazione o che a parità di età hanno la stessa dimensione
`-c`	salta i file che a destinazione hanno lo stesso checksum
`--exclude`	specifica path da non includere nella copia



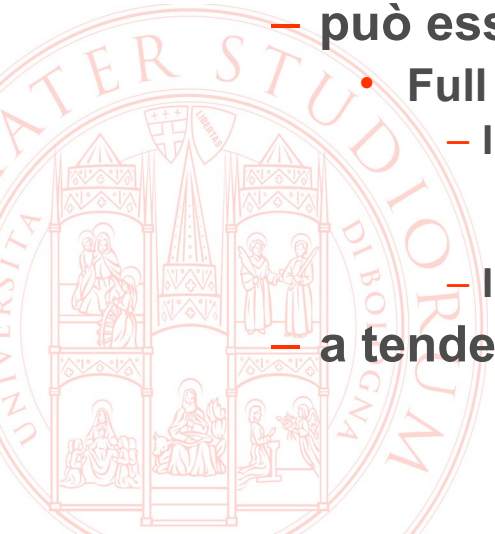
Backup

- Il backup è la copia dei dati dal sistema live ad un supporto offline
  - è impegnativo organizzativamente e tecnicamente
  - è l'assicurazione contro quasiasi causa di distruzione dei dati del sistema principale
  - Regola d'oro: 3-2-1-1 – 3 copie dei dati, in 2 differenti media, 1 copia offsite (cloud) e 1 copia offline
- Va pianificato, considerando tra gli altri questi fattori:
  - cosa copiare (compromesso tra praticità di ripristino e tempi/spazi necessari)
  - chi è incaricato dei backup
  - quando è necessario/possibile eseguire il backup
  - quanto rapidamente cambiano i dati sul sistema
  - quanto velocemente deve poter essere eseguito il restore
  - per quanto deve essere conservata ogni copia
  - dove saranno conservate le copie
  - dove saranno ripristinate le copie (compatibilità cross-platform)



Backup - strategie

- **FULL BACKUP** – è la copia completa di ogni singolo file nel/nei filesystem oggetto del backup
  - lento e ingombrante → difficile farlo frequentemente
  - massima semplicità di ripristino
- **INCREMENTAL BACKUP** – è la copia dei soli file cambiati da una data di riferimento, tipicamente quella di esecuzione dell'ultimo full backup
  - adatto all'esecuzione frequente
    - attenzione al carico della “semplice” operazione di indicizzazione
  - per il ripristino servono sia il full che l'incremental
  - può essere realizzato anche a più livelli
    - Full
      - Incremental/level0/volume1 (rispetto al full)
        - incremental/level1/volume1 (rispetto all'incremental/0/1)
        - incremental/level1/volume2 (rispetto all'incremental/0/1)
      - Incremental/level0/volume2 (rispetto al full)
  - a tendere, ad ogni cambiamento di un file → **point-in-time restore**

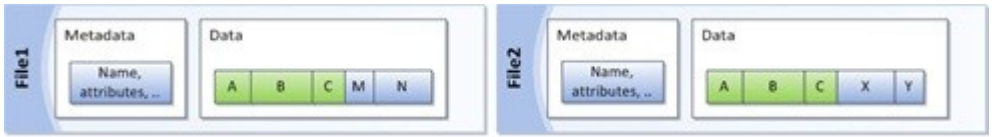


Backup - cautele

- **Correttezza della copia –** idealmente il filesystem dovrebbe essere a riposo durante il backup, ma è raro nella pratica, quindi bisogna curare bene i dettagli relativi alla lettura di file aperti o di strutture complesse come i database
- **Protezione dei dati –** un backup contiene tutti i file del sistema, ma **non c'è il sistema operativo a mediare l'accesso** quindi in caso di requisiti di riservatezza va difeso in modo diverso (fisico, cifratura)
- **Integrità dei dati –** se il backup viene svolto senza supervisione del sysadm, ci si deve cautelare da attività anche involontarie degli utenti che possano provocare la sovrascrittura dei dati
- **Affidabilità dei supporti –** con periodicità dipendente dalla criticità dei sistemi, ci si deve accertare che i dati siano scritti correttamente e siano leggibili per tutta la durata prevista della copia, curando
  - fattori tecnologici (graffi, smagnetizzazione, ***obsolescenza hw e sw***...)
  - fattori ambientali (polvere, umidità, temperatura, ...)
- **Facilità di reperimento –** i supporti devono essere organizzati per consentire di individuare facilmente ciò che si deve ripristinare

Backup – tecnologie

- Osservazione preliminare e comune a tutte le tecnologie fisiche: la computazione è a basso costo, conviene comprimere
  - ma molti dati sono nativamente compressi → pessimo rapporto tra carico di calcolo ed efficacia della compressione
- Soluzione allo stato dell'arte: *data deduplication*
  - Non solo per backup ma anche per main storage (es. ZFS)



  - Dataset diviso in *chunk*, identificati da un hash → se un chunk ha lo stesso hash di un altro, viene eliminato e sostituito da un puntatore

  - In teoria soffre del problema delle collisioni delle funzioni hash
  - In pratica la probabilità di una collisione è enormemente più bassa di qualsiasi altro errore nella catena di storage

Image source: https://technet.microsoft.com/en-us/library/hh831602(v=ws.11).aspx

Backup – tecnologie

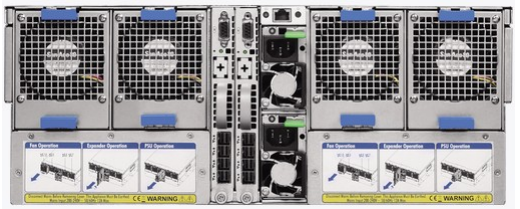
- Storicamente i backup venivano fatti su nastro già per sistemi di fascia medio-bassa
  - basso costo per byte
  - alta capacità
  - diverse soluzioni proprietarie ed incompatibili
- La crescita straordinaria della capacità degli hard disk ha messo in crisi le soluzioni tradizionali a nastro
  - è comune l'approccio disk-to-disk
  - per sistemi di fascia alta sono state sviluppate soluzioni a nastro estremamente performanti e con un alto costo d'ingresso, compensato dal basso costo marginale (per GB)
- I supporti ottici sono poco utilizzati su scala professionale
  - Limite: capacità
    - max attualmente disponibile (blue-ray XL) è 100GB
    - 1 HD da 20TB == 200 BDXL
  - Vantaggi
    - WORM (Write-Once Read-Many): i dati sono inalterabili una volta scritti
      - Affidabilità contro incidenti
      - Valore legale dell'archivio
    - Basso costo, lunga durata, semplice archiviazione
      - soluzioni domestiche
      - scenari in sono trattati relativamente pochi dati molto longevi

da non sottovalutare!

Qualche ordine di grandezza (2021/2022)

- Un esempio di storage server basato su HDD

  https://zstor.de/en/zstor-jbod-aj496-4u-96-bay.html
  - 96 HDD da 18TB = 1.7PB
  - varianti un po' meno sofisticate di pari capacità si trovano intorno ai 40k€+IVA = 3 cent/GB
  - 2kW sempre acceso!



- Cloud cold storage
  - es. AWS Glacier a Milano

    https://aws.amazon.com/s3/glacier/pricing/
  - zero investimento in costo capitale
  - uso: 0,4 cent/GB/mese + costi di retrieval

Qualche ordine di grandezza (2021/2022)

- Nastri LTO-9
  - tecnologia aperta (Linear Tape Open)
  - 18TB nativi
  - fino a 30 anni di durata in archivio
- Singolo tape drive
  - fino a 300MB/s sostenuti, 1200MB/s burst
  - compressione HW media 2.5x
  - cifratura HW
  - emulazione filesystem e integrazione con sistemi AAA e monitoraggio aziendali
- Tape library
  - nastri stoccati a consumo elettrico zero
  - fino a decine di migliaia di nastri = centinaia di PB → EB
  - investimento iniziale dell'ordine dei M€
  - costi su 5 anni per un Exabyte: 0,8 cent/GB
    https://blocksandfiles.com/2020/09/28/spectra-logic-exabyte-tape-library/
    https://spectralogic.com/how-to-store-an-exabyte/